ECE 376 - Homework #10

Timer1 Capture - Timer1 Compare.

555 Timer-Based Capacitance Meter

- Timer1 Capture:

Problem 1-5) Use Timer1 Capture to measure time to 1 clock (100ns). Some options are:

1) Requirements: Define the inputs, outputs, and how they relate.

- Part of the requirement must be to measure time to 100ns (i.e. use Timer1 Capture interrupts)

Measure a resistor with the period, T, measured with a resolution of 100ns

$$T = (R_1 + 2R_2) \cdot C \cdot \ln(2)$$

- R1 = 100k
- R2 = 100k
- C = 1uF (varies)

Computations

$$C = \left(\frac{T}{(R_1+2R_2)\ln(2)} \right) = 0.0000048090T$$

With T measured to 100ns

$$N = 10^7 T$$

$$C = 0.48089834N \qquad \text{pF}$$

If you capture every 16th edge

$$C = 30.056146685N \qquad \text{fF}$$

2) C-Code and flow chart.

Interrupts:

```
void interrupt IntServe(void)
{
   if (TMR0IF) {
      TMR0 = -10000;
      RC0 = !RC0;
      TMR0IF = 0;
      }
   if (TMR1IF) {
      TIME = TIME + 0x10000;
      TMR1IF = 0;
      }
   if (CCP1IF) {
      TIME1 = TIME0;
      TIME0 = TIME + CCPR1;
      N1 = TIME0 - TIME1;
      CCP1IF = 0;
      }
   }
```

Interrupt Initialization

```
// set up Timer0 for PS = 1
   T0CS = 0;
   T0CON = 0x88;
   TMR0ON = 1;
   TMR0IE = 1;
   TMR0IP = 1;
   PEIE = 1;
// set up Timer1 for PS = 1
   TMR1CS = 0;
   T1CON = 0x81;
   TMR1ON = 1;
   TMR1IE = 1;
   TMR1IP = 1;
   PEIE = 1;
// set up Capture1 for rising edges
   TRISC2 = 1;
   CCP1CON = 0x07;
   CCP1IE = 1;
   PEIE = 1;
```

Main Routine

```
   while(1) {

      fF = N1 * 30.056146685;

      LCD_Move(1,0);  LCD_Out(fF, 10, 6);
      SCI_Out(fF, 10, 6);
      SCI_CRLF();
      Wait_ms(500);

      }
```

3) Test: Collect data in lab to verify that your interrupts are working properly.

Toggle RA1 every Timer1 interrupt (2^16 clocks).

- Expected period = 2 * 65,536 = 131,072 clocks
- Measured period = 13.1063808ms = 131,063 clocks

Measure a 2ms square wave (555 timer with 0.36uF)

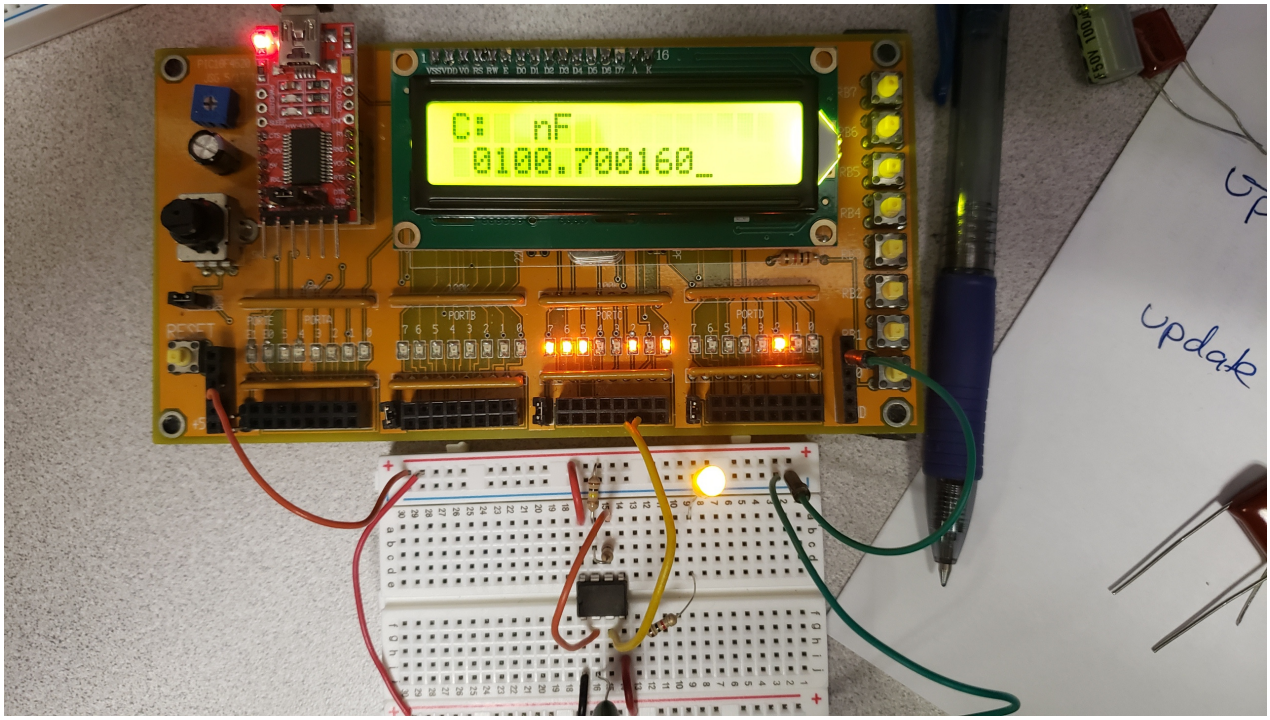
- Measured period = 1.7807872ms
- Calculated period = 1.8960*ms*

4) Validation: Collect data to validate your design works.

C	nF (measured)	
	mean	standard deviation
1uF	1036.752788	0.1772100908
0.1uF	100.6870528	0.014863695
0.082uF	79.106438095	0.012247297

note: both readings might be correct. C is specified at 1kHz. Our meter uses 23Hz - 13kHz.

5) Demo



Timer1 Compare:

Problem 6-10) Use Timer1 Compare to output precise frequencies. Some suggestions are:

- Precise 8 key piano: Play notes A3..A4 on RC0 when you press buttons RB0..RB7
- Electronic Trombone: Play note A3 (A2D=0) to A4 (A2D = 1023) on RC0 when you press RB0
- Input a number from 100 to 9999 with a keypad. Output that frequency on RC0
- Other...

Can you detect a 1% change in frequency at 440Hz?

6) Requirements: Press RB0 to start.

- The PIC flips a coin (head or tails)
- The PIC will then play 440Hz for 500ms
- Then pause 100ms
- Then play either 440Hz or 444.44Hz for 500ms, depending upon the coin toss (random).

The operator then must press a button

- RB0 if the notes sound like they're the same
- RB1 if the notes sound like they're different

The PIC then records whether you were correct or not, displays the running total on the LCD, the repeats.

7) C-Code and flow chart.

Interrupt Service Routine

```
void interrupt IntServe(void)
{
   if (TMR1IF) {
      TIME = TIME + 0x10000;
      TMR1IF = 0;
      }

   if (CCP2IF) {
      if(PLAY) CCP2CON ^= 1;
      CCPR2 += N2;
      CCP2IF = 0;
      }
   }
```

Main Routine

```
   while(1) {
      Wait_ms(1000);
      COIN = TMR1 & 1;
      N2 = 11364;
      PLAY = 1;
      Wait_ms(500);
      PLAY = 0;
      Wait_ms(200);
      if(COIN) N2 = 11364;
      else     N2 = 11250;
      PLAY = 1;
      Wait_ms(500);
      PLAY = 0;
      while(PORTB == 0);
      if(RB2) {
         RIGHT = 0;
         WRONG = 0;
         }
      else {
         if( RB0 ^ COIN) WRONG += 1;
         else RIGHT += 1;
      }
      LCD_Move(0,8); LCD_Out(RIGHT, 3, 0);
      LCD_Move(1,8); LCD_Out(WRONG, 3, 0);
      }
   }
```

8) Test: Collect data in lab to verify that your interrupts are working properly.

Test Code: Play 440.0Hz

```
while(1) {
   N = 11364; // 440Hz
   PLAY = 1;
}
```

Resulting frequency = 440.0Hz

Test Code: Play 444.44Hz

```
while(1) {
   N = 11250; // 444.44Hz
   PLAY = 1;
}
```

Resluting frequency = 445.0Hz

Test Code: Random number generator

```
while(1) {
   while(RB0);
   while(!RB0);
   DIE = TMR1 & 1;
   LCD_Move(0,0);  LCD_Write(DIE + 48);
}
```

Result

```
00100000111010110001010100010101111110100111110011101
```

- 25 0's
- 28 1's

9) Validation: Collect data to validate your design works.

- 18 tests
- Correct 15 times
- Incorrect 3 times

Guess	p	np	N	chi-squared
correct	0.5	9	15	4.00
incorrect	0.5	9	3	4.00
			Total	8.00

From StatTrek, a chi-scored critical value of 8.00 with 1 degree of freedom corresponds to a probability of 0.995

I can be 99.5% certain that I can hear a 1% difference in frequency at 440Hz (i.e. I'm not guessing)

10) Demo